

REMARKS

Claims 10-26 are pending in the present application. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 103, Alleged Obviousness, Claims 10-13 and 16-26

The Office Action rejects claims 10-13 and 16-26 under 35 U.S.C. § 103 as being unpatentable over Wallach et al. (Extensible Security Architectures for Java pages 116-128) in view of U.S. Patent No. 5,987,608 to Roskind. This rejection is respectfully traversed.

With regard to independent claim 10, the Office Action states:

As to claim 10, Wallach teaches a Privilege (page 120, line 9-24), Method (page 120, line 18-40), Data Processing System (NOTE: Data processing system will be inherent in the system of Wallach), a Current Thread (page 120, line 27-31), a Stack and Stack frame (page 120, line 32-54), a Thread Identifier ("....thread..." NOTE: It is inherent that the thread would have an identifier and since very thread is associated with a stack and stack frame, in order to access the stack the thread must provide some identifier that associates it with stack), locating a linked List and searching the linked list (Although, Wallach does not explicitly show a linked list, the algorithm searches a queue of frames that contains the enable privilege (page 120, line 1-14), a stack frame pointer (Wallach is not explicit about a stack frame pointer, however, the Target is an address/pointer to an address that matches the calling method (page 120, line 1-54). Wallach does not teach a run-time environment.

Roskind teaches a Run-Time Environment (Col. 4, Ln. 13-35). It would have been obvious to apply the teaching of Roskind to the system of Wallach. One would have been motivated to make such a modification to provide dynamic empowered access to security resources/method (Col. 2, Ln. 40-45).

Office Action dated November 7, 2002, pages 2-4.

Claim 10, which is representative of independent claims 19 and 26 with regard to similarly recited subject matter, reads as follows:

10. A process for providing a privilege for a method of a current thread that is currently executing in a run-time environment in a data processing system, the run-time environment having a stack comprising stack frames with stack frame pointers for associated methods, the process comprising the computer-implemented steps of:

using a thread identifier of the current thread, locating a linked list;
and
searching the linked list for an entry having a stack frame pointer
that matches the stack frame pointer of the method, wherein an entry of
the linked list is a stack frame extension.

Applicants respectfully submit that neither Wallach nor Roskind, either alone or in combination, teach or suggest locating a linked list using a thread identifier of a current thread or to searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of a method, wherein an entry of the linked list is a stack frame extension.

Wallach on page 120, which is the source of the alleged teachings relied upon by the Office Action, teaches a Java security mechanism in which privileges are enabled, disabled, and checked. Specifically, Wallach teaches that when code wishes to use a protected resource, such as a filesystem, the code must first call an `enablePrivilege()` method on the target associated with the resource. A policy engine will then be consulted to see if the principal of the code that called the `enablePrivilege()` method is permitted to use the resource. If permitted, an enabled privilege is created. After accessing the resource the code calls a `disablePrivilege()` method to discard the enabled privilege. The `checkPrivilege()` method searches for an enabled privilege to the given target. If one is not found an exception is thrown.

While Wallach mentions privileges, enabling and disabling privileges, and searching of enabled privileges, there is not one mention of a linked list, using a thread identifier to identify a linked list, searching the linked list for a stack frame pointer that matches a stack frame method for a method for which a privilege is to be provided, or entries in the linked list being stack frame extensions. To the contrary, rather than actually finding the features of claim 10 in Wallach, or some other reference, the Office Action merely makes assertions without any support from the cited references. For example, the Office Action asserts that "it is inherent that the thread would have an identifier and since every thread is associated with a stack and stack frame, in order to access the stack the thread must provide some identifier that associated it with stack." Applicants do not claim to have invented threads or thread identifiers. Applicants are claiming the use of thread identifiers to locate a linked list. There is nothing in the

general knowledge that threads may have thread identifiers that obviates using a thread identifier to locate a linked list that has entries that are stack frame extensions.

Furthermore, the stack is not equivalent to the linked list recited in the present invention. As is well known in the art, stacks are last-in-first-out based. That is, the last item, or address, placed (pushed) onto the stack is the first item removed (popped) from the stack. To the contrary, a linked list is a list of elements in which each element has a pointer to the previous element in the list. The stack is not a linked list. Therefore, even if it were inherent in Wallach, arguendo, to have a thread identifier that associated the thread with a stack, there is no teaching or suggestion to use such a thread identifier to locate a linked list.

In addition, claim 10 recites both a stack and a linked list. Therefore, the linked list is not the same as the stack since they are referenced as two separate elements in the claim. Thus, the stack in Wallach, again, is not the same as the linked list in claim 10.

The Office Action admits that Wallach does not teach a linked list but appears to equate "a queue of frames that contains the enable privilege (page 120, line 1-14)," allegedly taught in Wallach, with the linked list of claim 10. It should be noted that the section cited by the Office Action, makes no mention whatsoever regarding searching a queue of frames that contains an enable privilege. The only portion of Wallach that even discusses searching for a privilege is the description of the checkPrivilege() method which checks the frames on the stack of a caller of the checkPrivilege() method to determine if a stack frame has the requisite privilege to access a target resource.

As mentioned, previously, the stack is not the same as the linked list of claim 10. Moreover, the checkPrivilege() method does not search the stack to identify a stack frame pointer that matches the stack frame pointer of the method for which a privilege is to be provided. To the contrary, the checkPrivilege() method, as described in Wallach, searches the stack from the newest entry to the oldest looking for any stack frame that has the necessary privilege to access a target resource.

The Office Action alleges that the "target" is an address/pointer to an address that matches the calling method. However, if the target is the address/pointer to an address that matches the calling method, then why is the calling method checking to see if it has a sufficient privilege, through the calling of the checkPrivilege() method, to access itself?

This allegation by the Office Action makes no sense. While the target in Wallach may be the address of a target resource to which access is sought, the target is not a stack frame pointer of a method for which a privilege is to be provided. To the contrary, the target would be the address of the resource, e.g. filesystem, for which the calling method must have the requisite privilege to access. There simply is no teaching or suggestion in Wallach regarding the searching of a linked list, located using a thread identifier of a current thread, for an entry having a stack frame pointer that matches the stack frame pointer of the method for which a privilege is to be provided, as recited in claim 10.

Roskind does not provide for the deficiencies found in Wallach as noted above. Roskind is cited by the Office Action as teaching a run-time environment. There is no teaching or suggestion in Roskind with regard to any of the features that are missing in Wallach, as noted above. Therefore, any alleged combination of Wallach and Roskind, even if such a combination were possible and one of ordinary skill in the art were motivated to make such a combination, would not result in the invention as recited in claim 10.

Independent claims 19 and 26 recite similar features to those in claim 10 but in system and computer program product format, respectively. Therefore, claims 19 and 26 are considered to define over the alleged combination of Wallach and Roskind for similar reasons as noted above.

With regard to independent claim 18, the Office Action fails to address the specific features set forth in claim 18. Rather, the Office Action merely states "As to claim 18, see the rejection of claim 10." Claim 18 contains different features from those in claim 10 – features not addressed by the rejection of claim 10. For example, claim 18 recites "a set of stack frame extensions, wherein a stack frame extension comprises: a pointer to a stack frame for a method; a data field for privilege data for the method; a data field for validation data for the method." None of these fields of a stack frame extension are addressed by the Office Action and there is no showing where either of Wallach or Roskind allegedly teach these features. This is because there is no teaching or suggestion in Wallach or Roskind with regard to these fields of a stack frame extension.

In view of the above, Applicants respectfully submit that neither Wallach nor Roskind, either alone or in combination, teach or suggest the features of independent

claims 10, 18, 19 and 26. At least by virtue of their dependency on claims 10 and 19, respectively, neither Wallach nor Roskind, either alone or in combination, teach or suggest the features of dependent claims 11-13, 16-17 and 20-25. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 10-13 and 16-26 under 35 U.S.C. § 103(a).

Furthermore, neither Wallach nor Roskind, either alone or in combination, teach or suggest the specific features recited in dependent claims 11-13, 16-17 and 20-25. For example, neither reference teaches or suggests locating the linked list within a stack frame shadow apparatus comprising a plurality of linked lists, each linked list of the plurality of linked lists being associated with a thread, as recited in claim 11 and a similar feature in claim 20. With regard to this feature, the Office Action alleges that "since stacks are inherently made up of frames that linked the extended stack of Wallach would be made up of a plurality of linked list and the extended stack would also be contained in some type of apparatus." The Office Action is making allegations that are not supported by the reference. As noted above, a stack is not a linked list. Furthermore, there is no teaching or suggestion in Wallach regarding any stack frame shadow apparatus or even stack frame extensions. Moreover, there is no teaching or suggestion in Wallach to have a plurality of linked lists, each one associated with a thread.

Rather than pointing to any explicit teaching or suggestion in Wallach, the Office Action merely alleges teaches that are "inherent" in the reference. The Office Action is misapplying the doctrine of inherency. Under the principles of inherency, a claim is anticipated if a structure in the prior art necessarily functions in accordance with the limitations of a process or method claim. *In re King*, 801 F.2d 1324, 231 U.S.P.Q. 136 (Fed. Cir. 1986). Mere possibilities or even probabilities, however, are not enough to establish inherency. The missing claimed characteristics must be a "natural result" flowing from what is disclosed. *Continental Can Co. v. Monsanto Co.*, 948 F.2d 1264, 20 U.S.P.Q.2d 1746 (Fed. Cir. 1991). Unstated elements in a reference are inherent when they exist as a "matter of scientific fact". *Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 7 U.S.P.Q.2d 1057 (Fed. Cir.), *cert. denied*, 488 U.S. 892 (1988) and *Hughes Aircraft Co. v. United States*, 8 U.S.P.Q.2d 1580 (Ct. Cl. 1988). The features that the Office Action alleges are "inherent" are not a "natural result" or a "matter of scientific

fact” based on the teachings disclosed in Wallach. Rather, the allegation that these features are “inherent” in Wallach is an attempt to conjure up teachings in a reference that clearly are not there so that the reference appears to be more applicable than it really is. Wallach simply does not teach the features of the claims.

Regarding claims 12 and 21, these claims recite features similar to those features recited in claim 18 noted above. Therefore, claims 12 and 21 are also distinguished over the alleged combination of Wallach and Roskind because neither reference teaches or suggests a stack frame extension that comprises a stack frame pointer to a method, privilege information and validation information. The Office Action alleges that Wallach teaches validation information at page 120, lines 5-8. This portion of Wallach reads “As in the other approaches, extended stack introspection uses digital signatures to match pieces of incoming byte code to principals, and a policy engine is consulted to determine which code has permission for which targets.” While digital signatures are a form of validation information, there is no mention here, or anywhere else in Wallach, regarding a stack frame extension or the stack frame extension having a stack frame pointer to a method, privilege information, and validation information.

With regard to claims 13 and 22, the Office Action merely references the rejection of claim 12. Again, while Wallach teaches the use of digital signatures, there is no teaching or suggestion in Wallach to provide a stack frame extension that includes validation information that comprises the name of the method, the signature of the method, or the return address of the method.

With regard to claims 23 and 24, the Office Action states “see the rejection of claim 14” and “see the rejection of claim 15.” It should be noted that claims 14 and 15 are not rejected based on a combination only Wallach and Roskind. Rather, claims 14 and 15 are rejected based on Wallach, Roskind, and Introduction to the Capabilities Classes (see below). However, to the extent that the Office Action intended to reject claims 23 and 24 based on the alleged combination of Wallach and Roskind, neither reference, either alone or in combination, teaches or suggests adding means for adding an entry to the linked list if no matching entries are found in response to a request to enable a privilege for the method (claim 23) or removing means for removing a matching entry from the linked list if a matching entry is found in response to a request to revert a

privilege for the method (claim 24). The Office Action admits that Wallach and Roskind do not teach these features in the rejection of claims 14 and 15, discussed hereafter. Furthermore, as set forth hereinbelow, the Introduction to the Capabilities Classes reference also does not teach or suggest this feature.

Regarding claim 25, neither reference teaches or suggests retrieving means for retrieving privilege information and validation information for a matching entry from the linked list if a matching entry is found in response to a request to retrieve privileges for the method. As noted above, neither reference teaches a linked list and thus, cannot teach retrieving anything from such a linked list. Moreover, even the process in Wallach does not teach "retrieving" privilege information and validation information. To the contrary, Wallach teaches calling a `checkPrivilege()` method which consults a policy engine that informs the calling method as to whether there is a sufficient privilege to access a target resource. There is no retrieval of the privilege information or validation information from a linked list.

II. 35 U.S.C. § 103, Alleged Obviousness, Claims 14 and 15

The Office Action rejects claims 14 and 15 under 35 U.S.C. § 103 as being unpatentable over Wallach et al. in view of U.S. Patent No. 5,987,608 to Roskind as applied to claim 10 above, and further in view of Introduction to the Capabilities Classes (Hereinafter referred to as ICC pages 1-15). This rejection is respectfully traversed.

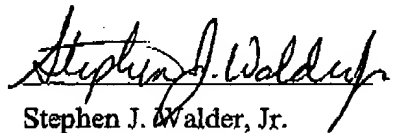
Wallach and Roskind have been discussed above. ICC does not provide for the deficiencies of these references. Specifically, ICC does not teach or suggest using a thread identifier of a current thread to locate a linked list and searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method, as recited in claim 10 from which claims 14 and 15 depend. Therefore, claims 14 and 15 define over the alleged combination of Wallach, Roskind and ICC. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 14 and 15 under 35 U.S.C. § 103(a).

III. Conclusion

It is respectfully urged that the subject application is patentable over Wallach, Roskind and ICC and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: February 7, 2003



Stephen J. Walder, Jr.

Reg. No. 41,534

Carstens, Yee & Cahoon, LLP

P.O. Box 802334

Dallas, TX 75380

(972) 367-2001

Attorney for Applicants